BITSZER USERS GUIDE 2022

BITSZER INCORPORATED | SUPPORT@BITSZER.COM

Table of Contents

INTRODUCTION:

Thank you for choosing to download Bitszer for your game. This asset has been developed to be as simple as possible to add into your game. All the heavy lifting of managing your gamers inventory has been offloaded allowing you to focus your time on what you do best, designing amazing games.

Even though Bitszer has been designed to be as simple as possible to integrate into your game, we do have tech support ready to answer your questions at support@bitszer.com

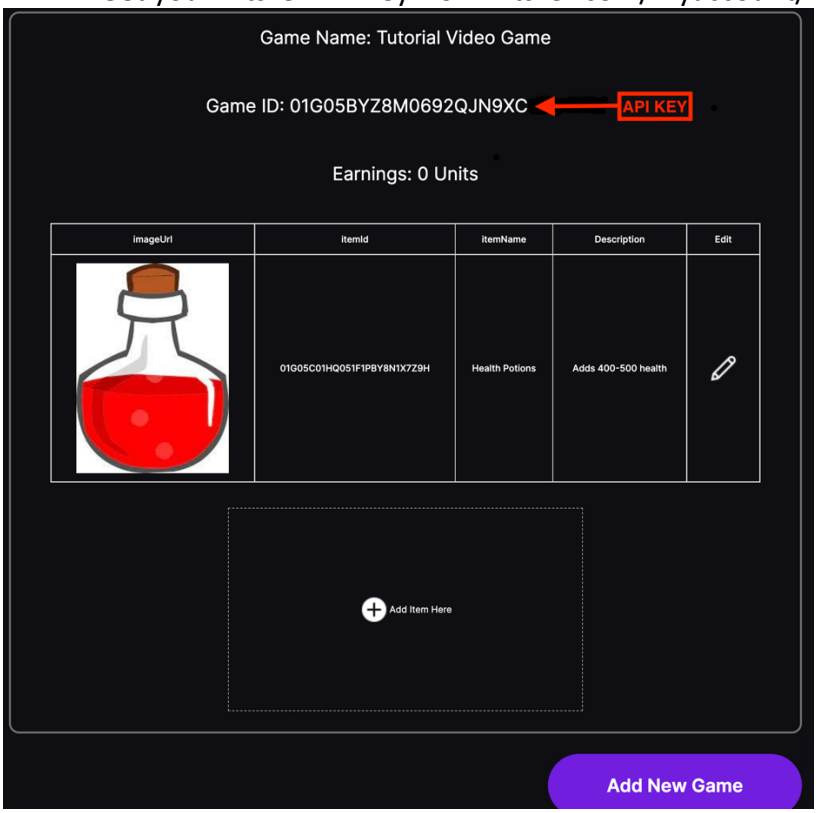
Bitszer comes with a preprogrammed UI to reduce development time. The UI can be changed and adjusted as you see fit.

Even though Bitszer maintains the servers for storing data as well as the API to get your inventory, Bitszer is free to use and to try. We never collect billing information.

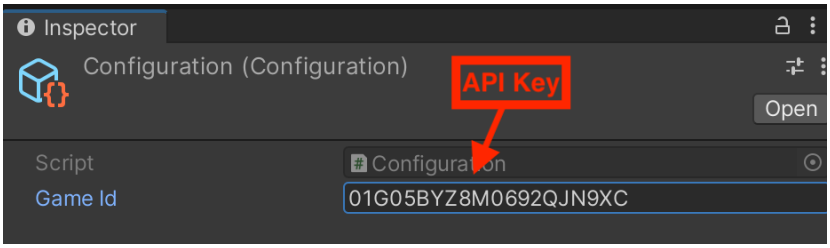
Consider this to be a living document that will be updated periodically.

QUICK START

- Import the asset from the asset store.
- Add the BitszerCanvas to your scene from _Project/Assets/Prefabs
- Get your Bitszer API key from Bitszer.com/myaccount/4



- Add you API key to the Configuration file, _Project/Configuration



Video: https://www.youtube.com/watch?v=GOVQ3PnogsA

OPEN AUCTION HOUSE

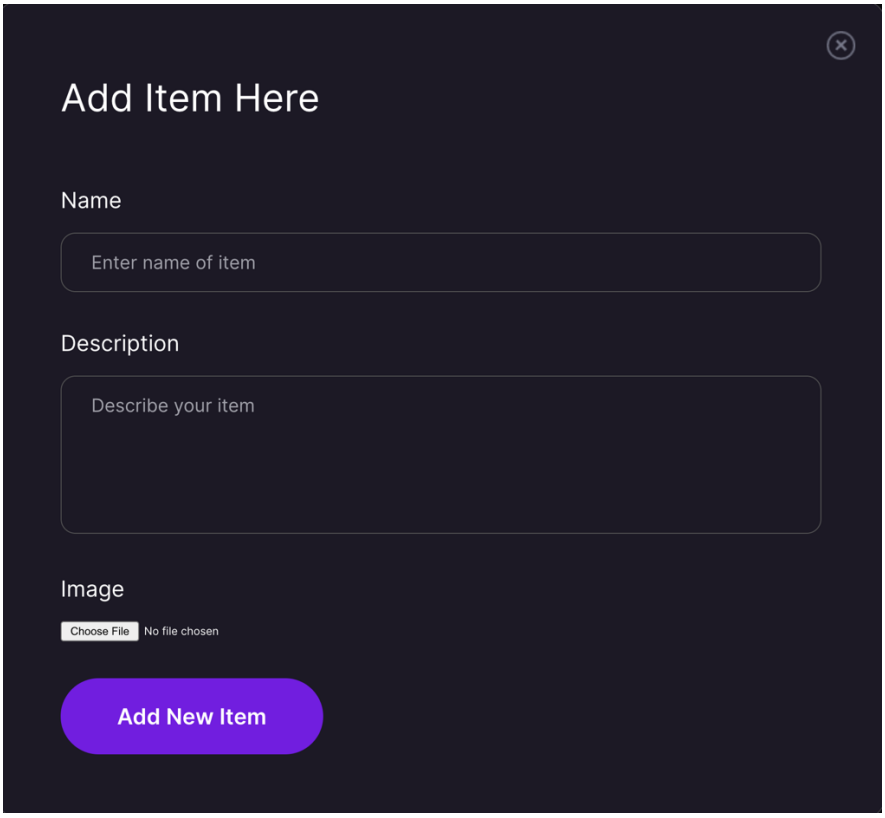
To open the auction house you need to create an Open Auction House function that you can call when an event triggers it.

```
using UnityEngine;
using Bitszer;

public class GameManager : MonoBehaviour
{    public void OpenAuctionHouse()
    {
        AuctionHouse.Instance.Open();
    }
}
```

ADDING INVENTORY

For all items that you want to be tradable on the auction house, you have to add the item through the inventory management system at Bitszer.com. Name, Description, and image can be edited at any time.



SAVING INVENTORY

When a gamer receives an item in the game, you want to save that to their inventory. This can be done by calling the Bitszer function “PushInventory.” First you create an array of InventoryDeltas to push. Each InventoryDelta consists of the itemId and the itemCount. Example:

```
public void AddValue()
{
    var currentValue = int.Parse(itemCountValueText.text);
    var valueToAdd = Random.Range(1, 10);
    var finalValue = currentValue + valueToAdd;

    itemCountValueText.text = finalValue.ToString();

    InventoryDelta[] inventoryDeltas =
    {
        new InventoryDelta
        {
            itemId = potionItemId,
            itemCount = finalValue
        }
    };

    StartCoroutine(AuctionHouse.Instance.PushInventory(inventoryDeltas, result => {}));
}
```

RETRIEVE INVENTORY

To get the inventory for the user we will use the Bitszer function “getMyInventorybyGame.” This method takes two arguments a limit and a nextToken. The limit is the number of items you want back and the nextToken is used to get the next page of items. For example, if you set the limit to ‘1’ but the gamer has two unique items then the function will return one item along with a nextToken. You can use the nextToken in subsequent calls to retrieve all items in the user’s inventory.

Example:

```
private void OnAuctionHouseInitialized()
{
  Debug.Log("AuctionHouse Initialized...");

  StartCoroutine(AuctionHouse.Instance.GetMyInventoryByGame(20, null, result =>
  {
    // Do whatever you need to do with the result.

    foreach (var item in result.data.getMyInventorybyGame.inventory)
    {
      if (item.gameItem.itemId.Equals("01G05C01HQ051F1PBY8N1X7Z9H"))
      {
        itemCountValueText.text = item.ItemCount.ToString();
      }
    }
  }));
}
```